12.0 ISSUES IN USER INTERFACE INTERNATIONALIZATION

Internationalization is the process of generalizing software so that it can handle multiple languages (i.e., locales) and cultural conventions without the need for re-design or re-compilation. Developers planning to field applications for use in combined or coalition warfare operations need to re-design the user interface so that it provides a "look and feel" that matches users' expectations, interacts with users in their native language, and displays data in a manner that is consistent with users' cultural conventions.

12.1 OPERATING WITH EXTENDED CHARACTER SETS

12.1.1 Character Rendering in Non-US Languages

Languages can be categorized in terms of the characters or symbols in which they are written. To facilitate computer processing, a character set is defined for each language to contain its written letters, numbers, and punctuation marks, with each character in the set represented by a binary value. Most European languages, including English, are based on the Roman alphabet. Because these languages contain fewer than 200 basic characters (i.e., the 26 letters in the alphabet, with upper case, lower case, and accented variations), their character sets can be encoded in a single byte. Single-byte character sets can be represented in a 16 x 16 matrix; each character is assigned a binary value ranging from 32 to 255, and the remaining values are reserved for computer control characters. ASCII (American Standard Code for Information Exchange) is the codeset in widest use in the US. All POSIX-compliant UNIX systems support this character set.

While most European languages are based on the Roman alphabet, many of them contain extended characters (i.e., ones that do not exist in English and are not available in ASCII) in their character sets. These characters include accented vowels such as é and ê; characters such as the French ç, the Spanish ñ, and the German ß and ü; and combined characters such as æ. In addition, some European languages may not use the entire Roman alphabet; Italian, for example, lacks the letter k. Despite these variations, all text in Roman-based languages is written from left to right, with each new character appended to the right of the previous character. Furthermore, the appearance of a character and its order within a character sequence do not change as new characters are entered.

While languages with fewer than 200 characters can be encoded in a single byte, complex languages such as Chinese, Japanese, and Korean each contain several thousand unique, ideographic symbols from which words are composed. Encoding such a large character set requires two bytes per character rather than one. Characters in these languages are represented by a mixture of single-byte and double-byte numeric codes ranging from 32 to 65,535. In multi-byte languages, there may be no "natural order" to characters vis-á-vis sorting, and no distinction between upper-case and lower-case forms. Because characters are composed of many strokes, text in these languages may require more space to display, and text entry may be a more complex process than in single-byte languages.

In most Roman-based languages, each character is rendered as a separate symbol of fixed shape, and characters are written in the approximate order in which they are pronounced. In other languages, however, the way in which characters are rendered graphically depends on their linguistic context. In Arabic, for example, a character may be displayed in several different forms, depending on whether it is displayed alone or as the first, middle, or last character in a word (see figure 12-1). In contextual languages, the characters that make up a symbol may be entered in several different orders, and entering a new symbol may change or even eliminate a previously entered symbol. As a result, complex algorithms may be needed to manage text line length (e.g., line breaks, justification), support the editing of individual characters within a symbol, and provide search and sort features that can recognize multiple encodings of the same symbol.

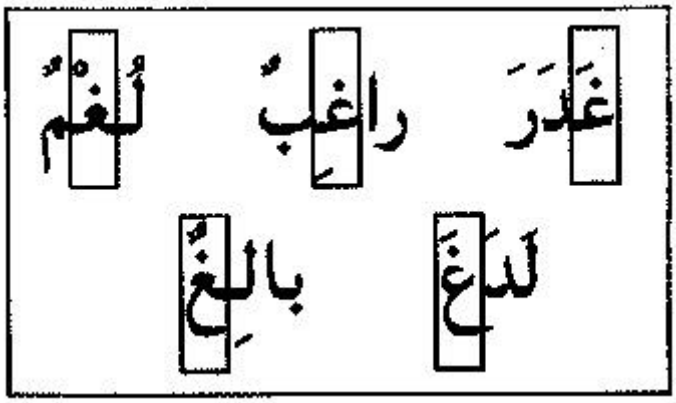

Figure 12-1. Forms of Arabic letter "G" (from Programming for the World).

Most languages are unidirectional; i.e., lines of text are presented uniformly from left to right or from top to bottom. Although Asian languages such as Chinese, Japanese, and Korean may present horizontal and vertical text on the same page, they are considered to be unidirectional because they do not mix directions in a single line of text. By contrast, Arabic and Hebrew are bidirectional; text in these languages is written from right to left, but numbers and foreign words in the same text are written from left to right. Because the direction of text entry may change from one character to the next, appropriate text handling procedures must be available in both right-to-left and left-to-right text.

12.1.2 Structural Rules for Character Handling

The application must be able to accept and process all of the characters in the character set used by the target language (i.e., the one to which the application is being converted). Because languages differ in their structural rules for character handling, assumptions made when processing a US character set may be inappropriate or inaccurate when applied to a language with extended characters. In particular, a US application is likely to require modification in order to correctly handle case conversion, ligatures, special characters, and word and character boundaries in the target language.

Case conversion. In US software, case conversion is usually performed by adding or subtracting a constant (i.e., 32) to or from the ASCII code for the character. In extended character sets, case conversion is more complicated because there is no constant difference between the numerical equivalents for upper- and lower-case representations of characters. In addition, the distinction made in English between upper-case and lower-case letters may be ambiguous or not exist at all in other languages. For example, Chinese, Japanese, and Korean have no case distinction. In other languages, an accented vowel in lower case may retain its accent in upper case, or the accent may disappear; alternatively, a vowel in upper case may or may not contain an accent in lower case, depending on the word and the language.

Ligatures. Ligatures are sequences of characters that are treated as a unit; e.g., æ is a combination of a and e. In some languages, ligatures can be entered as a single character or as two separate characters. In the latter case, both letters would be capitalized in words that are proper nouns; for example, Iceland is written as IJsland in Dutch. Ligatures occur frequently in context-dependent languages such as Arabic (as shown in figure 12-2). A US application may require revision in order to handle any ligatures that occur in the target language.

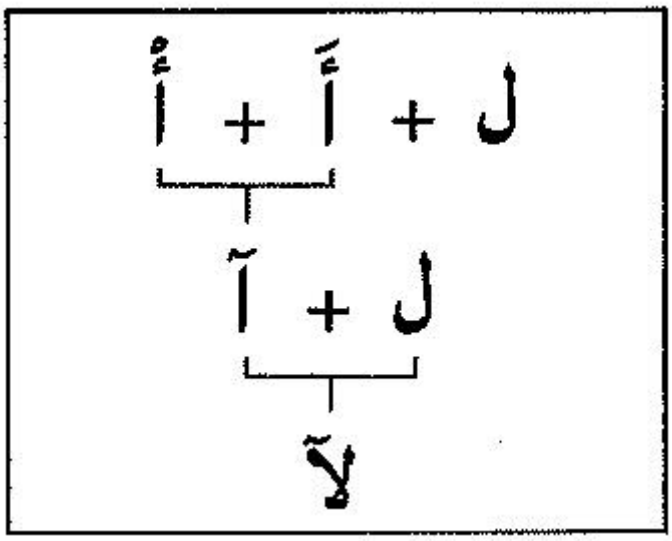

Figure 12-2. Example formation of Arabic ligature (from Programming for the World).

Special characters. Because languages differ in the meaning assigned to special characters, a US application that uses characters such as apostrophes as delimiters in a text string and restrict their use to this function may require modification when converted to certain European languages. For example, French and Italian replace the terminal vowel in an article by an apostrophe when the following noun has an initial vowel. In addition, some languages include special characters that may not be present in English or use these characters in ways that differ from US usage. For example, Spanish starts exclamations and questions with inverted exclamation mark and inverted question mark characters, while French includes a space between the last word of a sentence and a concluding exclamation or question mark. Finally, diacritical marks (i.e., the signs modifying the value or sound of characters) may have different meanings in different languages. For example, certain diacritical marks specify the doubling of consonants in Arabic but may indicate pitch in Vietnamese.

Word and character boundaries. A word in text consists of a string of characters between delimiters. In a US application, these delimiters are usually blanks or spaces but may also include the unused portion of the Roman character set. This latter approach can be problematic when converting to a Roman-based language with extended characters. These characters need to be interpreted as part of the word and not considered as delimiters. In addition, in some languages, a blank is acceptable as a numeric or phrase separator and so would not be appropriate to use as a standard word delimiter.

12.2 TEXT TRANSLATION

12.2.1 Creating Internationalized English Text

The process of text translation begins with the creation of an “internationalized English” version of the US application. All text displayed by the application is reviewed and, if necessary, modified to ensure that it is easy to understand and use. Message text (e.g., in message windows, online help) is presented in short, simple, declarative sentences whenever possible. Excessive use of subordinate and coordinating phrases is avoided, and ideas are expressed as concretely as possible. Ambiguous language, humor, jargon, and cryptic messages are likely to cause difficulty for non-US audiences and so need to be eliminated. Likewise, compound adjectives, strings of nouns, long sentences with many ideas, and negative questions can be difficult to understand and are not used. If the application needs to explain a series of concepts, they are presented in the form of a list, rather than in a text string separated by commas.

The content of each window in the application needs to be checked for US-specific language prior to translation to minimize the likelihood of misinterpretation by the target audience. The goal is to use only those terms that are employed in the same way throughout the English-speaking world. The use of acronyms and abbreviations is limited since many are not recognized internationally and may have different meaning, depending on where they are used. Similarly, when large numbers are presented, they are written as numerals; for example, the term "billion" means one thousand million in the US but one million million in some European countries. To minimize confusion, the names of months are written out when they appear as part of a date. For example, 06/10/94 can be interpreted as June 10 or October 6, depending on the user's experience.

The application avoids presenting examples that may be uniquely American. A generic term is used, rather than what something is called in the US. For example, "stock exchange index" is an international term while "the Dow" is specifically American and "the FTSE 100" is specifically British. The messages in the application are reviewed to determine if they may be interpreted by the target audience in ways other than intended. For example, "as soon as possible" means "immediately" in the US but "when convenient" in other countries. Terms such as "left hand" can be offensive in some cultures and need to be replaced with "on the left" or "left side" instead.

12.2.2 Translating Text and Messages

All text displayed by the application (e.g., text in windows, alerts, messages, help) is translated into the target language. If appropriate, the translation is tailored to the target language in the specific country or region that will be using the application. Translated text contains proper technical terminology, especially when the terminology may differ from conversational expression in the target language. Care is taken to maintain distinctions between terminology that may be translated into the same text string in the target language. For example, Cancel and Undo are normally translated as "annulez" in French even though these commands have distinct meanings in English. Similarly, spelling and grammar can differ among varieties of a single language and so need to be adapted accordingly.

The same terminology is used in both the user interface and documentation for the application. While the goal is to provide an accurate translation of all text into the target language, it is acceptable to use US words in the text if the language does not have an adequate vocabulary with specific translations of technical words or if the target audience is accustomed to dealing with US terminology.

The accuracy of translated messages is verified since it is possible for the text of one message to be the same as another, especially if the original messages were very similar or were worded ambiguously. In addition, the meaningfulness of the translation is checked against the situation that invoked it to ensure that the information being conveyed in the original message is also conveyed in the translated version.

Translated text is reviewed to ensure that it makes grammatical sense. Some applications construct messages from two or more substrings; for example, the name of the file being deleted is inserted into the text string requesting the user to confirm deletion of the file. While this approach may work in English, the linguistic characteristics (e.g., gender, word order, special characters) of the resulting text can be awkward or inappropriate when translated into the target language. When messages are constructed by nesting or concatenating strings, the translated text that results is frequently meaningless or syntactically impossible because words or phrases were not modified to fit the grammatical rules of the target language.

Translated text uses the same character set and font as the rest of the application. Line breaks and other format changes that may have been introduced with the translation are checked for accuracy since US rules for hyphenation, punctuation, or capitalization are likely to be different from those in the target

language. If typographic variations such as italics or boldface have been added as part of the translation, they are checked to ensure that they are suitable in the target language.

Language environments have evolved unique rules defining how elements such as title lines, bulleted lists, and footnotes are used to distinguish among levels of expression and to indicate how expressions are related. The appearance of translated text is adapted as needed to satisfy these rules.

12.2.3 Translating Documentation

The documentation for an internationalized application describes a representative sample of the internationalized capabilities provided by the software. For example, an explanation of how a sorting function works describes the kinds of sorts that are performed, explains that the current locale affects the output, and provides several examples that are representative of the locales supported.

Documentation text contains simplified English. Whenever possible, a single term is selected to express a concept, and the use of synonyms is minimized. However, these changes are made in such a way as to not reduce the precision of the text, create awkward phrasing (with an increase in overall text length), or produce unacceptably dull or boring text. References (e.g., to sample users) and examples that are specific to one culture are modified to be more international in focus. Any graphic symbols used in the documentation are reviewed to minimize the extent to which they are culture-specific. If necessary, a table is provided that lists the symbols and their interpretation. Finally, documentation sections such as glossaries and indexes is expanded as needed to help non-US readers find information. For example, glossaries define words that may have a different technical meaning or not exist at all in the locales supported by the application.

Because the order of items in a sorted list usually changes following translation, references to the position of items in the list are removed from documentation. Similarly, when collation sequences are described, the results are not described as sorted "alphabetically" since ideographic languages cannot be sorted alphabetically. Output is described as appearing "in sorted order as determined by the current locale." Other changes needed when internationalizing documentation include ensuring that terms such as ASCII, text, byte, and character are used appropriately, describing any assumptions made about date and time formats, replacing references to Yes and No (e.g., when describing actions in response to a message) with words that are appropriate to the locale, and presenting the names of any individuals (e.g., sample users) in an order that is correct for the specific culture. Finally, lengthy text explanations (e.g., in online help, training materials, or other documentation) may need to be restructured or reorganized so that they follow the rules used by the target audience in organizing technical discussions or sequences of explanations.

12.3 TEXT INPUT METHODS

12.3.1 Keyboards and Keyboard Input

A language usually has a specific keyboard layout associated with it, and this layout may be different from the one available on US workstations. Conventions concerning the location of characters vary from language to language and sometimes from country to country within the same language. For example, the German keyboard reverses Z and Y from their positions on the US keyboard, and the Spanish keyboard has a different layout in Spain than in Latin America. In addition, languages may add, omit, or change the characters on a keyboard. For example, the British keyboard contains the currency symbol for pound instead of #, and the Spanish keyboard has ñ where the US keyboard has L. Non-US keyboards may mark each key with up to four different characters. Users press modifier keys (e.g., <Shift> and/or <Alt>) in combination with the key to enter the various characters marked on the key.

Because computers respond to specific physical keypresses regardless of what markings appear on the keys, a different keyboard may not be required when converting a US application into another language. A keyboard can be adapted by replacing the symbols on each key, either with adhesive labels or new key covers. The application then maps the individual keystrokes to the character set for the other language and displays the appropriate characters on the screen. If this approach is used, the function keys on the keyboard need to be mapped to the same actions as in the original software, and any messages generated when these keys are pressed are displayed as they were prior to the conversion.

Languages where diacritical marks are used extensively (e.g., French) usually provide keyboards that allow users to generate characters with these marks with a single keystroke. However, because English has very few accents, users with a US keyboard have to execute a combination of keystrokes in order to enter an extended character. "Dead" keys and a compose-based method can be implemented in the application so that users can produce this type of input.

With "dead" keys, the keystrokes consist of a "dead" (i.e., nonspacing) key, followed by the character (e.g., a vowel) to be displayed with an accent. A different dead key is assigned to each accent. When a dead key is pressed, a text input mode is invoked; the symbol on the key is not displayed, and the text cursor does not move. The mode is automatically disabled following the next keystroke; the appropriate dead key is pressed each time an accented character is being entered. If an invalid character (e.g., a consonant) is entered, the character is displayed without an accent, and feedback (e.g., a beep) is provided to indicate that the keystroke was invalid.

In a compose-based input method, when a predefined control key is pressed, a text input mode is invoked that forms the next two keystrokes into a single character. When the first character (e.g., a vowel) is typed, nothing is displayed on the screen. When the second character (e.g., the diacritic) is entered, the completed character is displayed, and the input mode is automatically disabled.

12.3.2 Approaches to Text Entry

Pre-edit methods of text entry. In most languages, users perform text entry by typing directly into a text field. However, if a keyboard cannot produce all of the symbols in a target language, a pre-edit step may be needed. Users type characters from the keyboard, usually into a pre-edit area, and then execute an action to convert the characters into other symbols appropriate to the language. These symbols are then displayed in the text field.

When a pre-edit step is required, text entry can be performed on-the-spot, over-the-spot, or off-the-spot. On-the-spot means that as users type, the characters appear directly in the text field which can contain both text in unconverted form and converted symbols. Although more difficult to implement, this approach is preferred because it is more similar to text entry as normally performed by users. In over-the-spot, a separate pre-edit area is provided for each text field; when users convert their input into final form, the symbols are displayed in the appropriate text field. Off-the-spot also provides a separate pre-edit area but uses the same area for multiple text fields; in this case, when users convert their input into final form, the symbols are displayed in the text field that has input focus.

When text entry includes a pre-edit step, the application provides feedback concerning the status of the input after users enter text in the pre-edit area and then execute an action to convert the input into final form. If insufficient information is available to perform the conversion, the application can prompt users to enter more pre-edit text, present them with a list of choices from which to select, or indicate that the conversion has failed. If an on-the-spot approach is implemented, the text field provides a visual distinction (e.g., a different text font or color) between original input and converted text so that users can easily distinguish between the two. If the pre-edit area is provided in a separate dialog window, the window is modeless so that users are not restricted to only performing text entry.

Text entry in languages with large character sets. Several options are available to support keyboard input in languages such as Chinese, Japanese, and Korean that have large character sets. Whatever method is selected must be able to accommodate context-specific variations within the language as users perform text entry. With each keystroke, converted text changes as needed in order to create a new compound character or add a mark to a previous character.

With the first option, the component elements of each character are marked on the keyboard. As users press individual keys, the elements are displayed. When a character is complete, it is displayed in place of its components. This method has been used to perform text entry in Chinese and Korean.

With the second option, users enter each character phonetically, and the phonetic form is automatically translated into the correct character. When more than one character has the same pronunciation, users are presented with an array of phonetically similar characters from which to choose. For example, users enter a root or radical character from the keyboard, then select additional strokes to complete the character from a set displayed on the screen. This method has been used to convert Roman characters to Chinese ideographs, and Hiragana and Katakana characters to Japanese Kanji.

With the third option, users type the decimal or hexadecimal encoded value for a character or select a value from a list. If the value matches an entry in the code set, the corresponding character is displayed on the screen.

Text entry in mixed character sets. Users may need to perform text entry in more than one character set (e.g., English and Korean) or in multiple locale-specific character sets (e.g., Kanji and Katakana). This flexibility can be provided by defining text input modes in each character set, along with a special keyboard character that allows users to toggle back and forth between the character sets as desired. Users with a keyboard where two character sets are marked on the keys select one of the modes to begin text entry. All of the typed text is interpreted in this character set. When the special character is encountered, the text mode toggles to the other character set and all subsequent input is interpreted in this set.

Text entry in bidirectional languages. Because bidirectional languages write text in both right-to-left and left-to-right directions, text entry may be performed in either direction, depending on the contents of a text field, and may require input in both directions within a single field. The application can support bidirectional text entry through the use of a text input mode that switches the direction of text entry when invoked. Users typing right-to-left text in a right-to-left text field press a special keyboard character to invoke the mode and type left-to-right text (e.g., a number) in the field. During left-to-right text entry, the text cursor remains at the boundary between the two text segments, and the left-to-right text is displayed to the left of the cursor as it is entered. Users press the key controlling text input mode a second time to shift back to right-to-left text entry. When the mode is disabled, the text cursor jumps to the left end of the left-to-right text that was just entered, and text entry proceeds again in a right-to-left direction.

12.3.3 Other Text Entry Actions

The text cursor remains visible during text entry to indicate the locus of typed input. In addition, the text cursor does not disappear from view as it moves from one character to the next in a string of single-byte and multi-byte characters. If the target language is bidirectional, the application can support multiple text cursors within a single text area, one indicating when text can be added in the current input direction and the other marking the last place where the direction of input changed. In contextual languages (i.e., where the appearance of existing text can change as new characters are entered), the text insertion point can move backward or forward as users perform text entry; in this case, the text cursor is displayed in a manner that is consistent with the movement of the text insertion point.

The arrow keys move the text cursor in the direction of the arrow regardless of the direction in which text is currently being entered. <Delete> deletes text in the direction opposite to the direction in which text is being entered.

If the application is being converted to a contextual language, it needs to define how certain keystrokes affect a compound symbol that is composed of several separate characters. For example, the application needs to determine when <Delete> cancels the previous keystroke (i.e., removes a character) or deletes the entire symbol. In addition, the application may need to limit the ability to insert or delete individual characters in a word if these actions would change neighboring characters or alter the appearance of the word in unintended or confusing ways.

12.4 INTERNATIONALIZING USER INTERFACE FEATURES

12.4.1 Text Expansion

When English text is translated into another language, the result is often longer than the original English. For example, the phrase "message popup" translates to "Nachrichtenüberlagerrungsfenster" in German and "janela de sobreposiçao de mensagem" in Portuguese. The increase in text length may be as much as 200 percent, depending on the length of the original text. Some of this increase may result from the addition of spaces that were not present in the original text. Table 12-1 lists recommended allowances for expansion based on text length in English. This table refers to the number of characters in a message, with characters in multi-byte languages (e.g., Japanese) taking two bytes per character.

Table 12-1. Allowances for text expansion.

Length of English Text	Additional Space Required
Up to 10 characters	101 - 200 percent
11 - 20 characters	81-100 percent
21 - 30 characters	61 - 80 percent
31 - 50 characters	31 - 40 percent
51 - 70 characters	31 - 40 percent
Over 70 characters	30 percent

Note: This table was taken from the Microsoft Windows Software Development Kit -- Additional Windows Development Notes, as published in Software Internationalization and Localization: An Introduction.

Translated text may require adjustments in the horizontal spacing between specific pairs of characters. For example, in English, the characters f and i look better when displayed closer together than other pairs of characters. The horizontal spacing algorithms used by the application need to accommodate adjustments in the spacing of non-US characters, including pairs of characters (e.g., ₧) that may be part of an extended character set.

The height of a line of translated text may be twice the height of the original text in English. Roman-based languages may supplement the character set with diacritical marks that extend above or below the basic symbol. In non-Roman languages, marks may be stacked two or three high, and small versions of characters may be placed above, below, or beside the primary symbols, causing wide variations in the height of each text line. Because of their complexity, ideographs require more space to display the strokes within them. For example, some complex Chinese characters may need to be displayed at least 50 percent larger than alphabetic characters in order to be readable. The minimum size for ideographs is

usually 16 x 16 pixels. Translated text may also require adjustments to the vertical spacing between lines to ensure legibility and readability both when displayed on the screen and when printed. Extended characters, and in particular those with diacritical marks, may require additional spacing, especially when printed in upper case.

It is likely that the size and placement of controls in application windows will require adjustment in order to accommodate text expansion following translation. Menus and dialog windows will also need to increase in size in order to accommodate the longer text. Similarly, more vertical space may be needed in window components such as the title bar to accommodate larger character size, especially in languages such as Chinese, Japanese, and Korean. Finally, the size of the text included in the label of a window icon may need to increase to accommodate an extended character set, and the icon graphic may also contain embedded text that needs to be translated.

Each application window needs to be checked to ensure that all of the translated text fits properly within the window and that individual controls are positioned correctly within each window area. For example, when column headings are translated, the text may be longer than the data included in the column so that the heading has to be broken into more than one line of text. Similarly, when text labels are translated, the placement of the associated text fields may be altered and require repositioning in order to be properly aligned within the window.

12.4.2 Nonlinguistic Text Features

Capitalization, punctuation, and word order. When text is displayed in application windows, it follows the rules for capitalization, punctuation, and word order used in the target language. For example, in German, all nouns are capitalized, regardless of their position within a phrase or sentence. Depending on the language, quotation marks may be displayed as “quotation”, «quotation», »quotation«, or ,,quotation.“ Interrogatory sentences in Spanish begin with an inverted question mark and end with a question mark in normal orientation. Adjectives precede nouns in English word order but may follow nouns in other languages.

Hyphenation. Hyphenation is performed by the application in a manner that is consistent with the rules of the target language. These rules may call for changing the characters in a word when it is hyphenated at the end of a line, or placing hyphens between individual words when they extend beyond the end of a line. For example, in German, “drucken” and “heißen” become “druk-ken” and “heis-sen” when hyphenated. In French, a hyphen is added between a personal pronoun and “mê me” (e.g., “eux-mê me”) when these words extend beyond the end of a line.

Justification. The justification routines used by the application conform to the rules of the target language and may require some character-processing logic in order to do so. For example, in Asian languages such as Japanese where spaces are not used to delimit words, line breaks can occur anywhere within a word. However, because symbols are represented by a multi-byte character, line breaks cannot occur within a symbol nor can punctuation be the first character on a new line. Alternatively, other languages such as Arabic and Hindi do not allow breaks within words. In this case, the justification algorithm used by the application must accommodate this restriction and be able to produce justified text without excessive space between words.

Abbreviations. The abbreviations used in US software may have different meanings in other languages or not be used at all. For example, while # is commonly used as an abbreviation for number, this character is not meaningful outside the US. The symbol @ means “at” in the US but “each” in the United Kingdom. The abbreviations for ordinals are 1st, 2nd, 3rd, etc. in the US, but 1^{o}, 2^{o}, 3^{o} or 1^{a}, 2^{a}, 3^{a} in other languages, depending on the gender of the subject.

Typography. If the application displays text in a Roman-based character set, it supports the fonts (e.g., Times and Helvetica), sizes (e.g., 10-point, 12-point), and styles (e.g., plain, italic, bold) that are normally available in the target language. The application also accommodates any unique typographic conventions when displaying translated text. For example, stress in writing is indicated through the use of italics in English but by letter spacing or boldface in European languages. In Japanese, stress is indicated by underlining characters, putting a light gray background behind them, or writing the text in Katakana.

Reordering sorted information. Applications frequently present sets of related items (e.g., in lists, option menus) in alphabetical order. These items need to be reordered after translation according to a sorted order that is meaningful to the target audience. The most appropriate order may vary by application and depend on the information displayed in the items.

12.4.3 Data Formats

The application is able to recognize and correctly handle the range of formats that are used to express data in the target language. The labels for all data fields are modified to include the unit of measurement required for data entry. The application either converts the data format to one that is familiar to users or provides the capability to display data in alternate formats so that users can select the one that is most meaningful to them. For example, US users prefer to measure length in feet and yards while European users are more familiar with the metric system. If the content of data fields is not converted to a data format that is familiar to the target audience, then the label for the field is extended to include the data format as part of the label. If the application supports converting to and from both millimeters and inches, the number of digits stored is sufficient to prevent truncation errors during conversions.

The presentation of date and time is modifiable by users so that they can display this information in the appropriate time zone (e.g,, India rather than Zulu) and modify it for other zones as needed. Numeric data is properly aligned according to the particular numerical separators and indicators used in the target language. In addition, if the application allows users to manipulate text, the different forms of tabulation available are modified as needed (e.g., allow the decimal tab to work with commas rather than periods) to accommodate the data formats used in the target language.

Number systems and formats. While Arabic numerals (e.g., 0, 1, 2, etc.) are widely accepted, some languages have their own numbering systems. In some cases (e.g., Chinese), the symbols are substitutes for Arabic numerals, while in others (e.g., Ethiopia), there are special characters for numbers such as 10 and 100.

When presenting numbers, a comma, period, space, and apostrophe can be used as separators for units of thousands. In some cases, an explicit separator is not required for numbers less than 10,000. Numbers can be grouped by thousands or ten thousands. The period, comma, and center dot can be used as separators for decimal numbers. Positive and negative numbers can be indicated by + and - symbols appearing either before or after the number, and negative numbers can be enclosed in parentheses.

Measurement systems and arithmetic operations. US users are familiar with the Imperial system of measurement in inches and fractions of an inch (e.g., halves, quarters, eighths) while users outside the US rely on the metric system which uses meters, liters, and grams. In addition, the US relies on the Fahrenheit scale for temperature while the rest of the world uses the Celsius scale. Similarly, traditions vary in the manner in which certain arithmetic operations are performed. For example, some countries have rules for rounding numbers that differ from those used in the US. In addition, accounting rules (e.g., to calculate compound interest) vary from locale to locale.

Currency. The comma, period, and colon can be used as separators for currency. Currency indicators include a number of symbols (e.g., $, British pound, and the Japanese yen), alphabetic characters (e.g., FF, SFRs, kr), and combinations (e.g., CZ$), and can be placed at the beginning, middle, or end of the currency expression. There can be one or no space between the currency symbol and the amount, and currency symbols can be up to four characters in length. Most currencies (except Japan) include two digits to indicate fractional money amounts.

Date and time. The hyphen, comma, period, space, and slash can be used as separators for the day, month, and year, or separators can be left out altogether. In numeric date formats, the month and day fields can be reversed, and in some cases, the year field can come first. Month and day names can be capitalized or in lower case and can be abbreviated using the first two or three letters or some other combination of letters.

The manner in which a date is expressed can be affected by the calendar system being used. While dates are usually based on the Gregorian calendar, some cultures use lunar calendars or the Jewish or Arabic calendar or can express the date based on the year of accession of the Emperor, as in Japan. These calendars can include day names for more than seven days, and month names for more than twelve months. Moslem countries such as Saudi Arabia and Egypt use a calendar with 12 months but only 354 or 355 days. The first day of the week is Sunday in the US but Monday in European countries, a difference that affects the manner in which calendars are displayed.

The colon, period, and space can be used as separators for hours, minutes, and seconds. The letter h can separate hours and minutes. Both 12-hour and 24-hour notation can be used. For 12-hour notation, a.m. or p.m. can appear after the time.

Although the world is divided into 24 standard time zones, countries have the freedom to set their own times. For example, in South America, Surinam's time is 30 minutes different from that of the next zone, and Guyana's is 45 minutes different. The same time zone can have multiple names, and different time zones can share the same abbreviation. Finally, countries differ in their rules concerning daylight savings time or may not use it at all, and the hemispheres differ in when it starts and ends because the seasons are reversed.

Addresses and telephone numbers. Addresses vary from two to six lines long and can include any character used in the character set for a language. The house number precedes the street name in the US and United Kingdom but follows the street name in most other European countries. Postal codes appear in various positions and can include alphabetic characters (e.g., an abbreviation for the country), separators (usually spaces), and numbers (up to seven characters and numbers in length). In many countries, each part of an address is written on a separate line; however, in South Korea, the entire address is placed on a single line, with the specific format used varying for central cities and local areas.

Telephone numbers can contain blanks, commas, hyphens, periods, and square brackets as separators. Telephone numbers can be displayed in local, national, and international formats. Local formats vary widely. National formats can have an area code in parentheses, while international formats can drop the parentheses but add a plus sign at the beginning of the number to indicate the country code.

12.4.4 Graphics

Icons and symbols. The graphic features (e.g., icons) of an application designed for a US audience may appear strange when viewed by users outside the US. For example, a mail application that changes a mailbox graphic to indicate receipt of new mail may be unrecognizable in another culture where mailboxes have a different appearance or may not be used. Certain images, colors, and numbers of objects in a group may evoke a negative reaction in another culture so that they obscure or contradict

the message they are intended to convey. As a result, the icons used in the application may need to be revised in order to match the image or symbol to the culture in which the application will be used.

Application icons use existing international symbols whenever possible. When a new symbol is created, it represents a basic, concrete concept because concrete icons require less explanation than abstract ones. In addition, each new symbol needs to be compared with existing symbols to ensure there are no conflicts. The use of stars and crosses as part of the symbol is avoided. Text is not included in an icon because it will need to be translated and may not fit into the icon when presented in the target language.

Drawings. Application windows presenting information in graphic form (e.g., line graphs, bar charts, histograms, flowcharts) incorporate translated text, including appropriate adjustments in data formats as needed. The size of the graphics objects may need to be enlarged to accommodate the increased length of translated text. Alternatively, application graphics can be modified to place text adjacent to, rather than within, the object so that changes in text length do not affect size of individual objects or the overall illustration.

Graphic design conventions vary from culture to culture. For example, Japanese artists tend to draw tables of data differently than Western artists do. As a result, the application may require modification to accommodate these conventions.

Tactical graphics. When tactical graphics are presented (e.g., in a map window), users can access a variety of map features in order to customize the display to match their preferred mode for viewing and interpreting this information. For example, where US users are likely to display road features for navigation in urban areas, Korean operators may prefer to see neighborhood names as key map landmarks.

Visual cues for alerting. There are cultural differences associated with the meaning of certain graphics used for alerting (e.g., gestures such as waving one's hand) that may be offensive to members of the target audience. The specific visual signals used by the application, especially for alerting, need to be reviewed to ensure that they convey the desired meaning in the target culture and that their representations within the software will not be objectionable to users. In addition, alert and warning messages can be supplemented with icons so that the application communicates critical information in both text and graphic form.

12.4.5 Keyboard Interaction

Mnemonics and accelerators. When menu options are translated, any mnemonics or keyboard accelerators included with the options also need to be modified so that they contain letters that reflect the translated text. In general, the guidelines for mnemonics in languages with single-byte character sets also apply to languages with multi-byte character sets, except for how mnemonics are displayed. Applications translated from the former to the latter can retain the mnemonics used in the single-byte version, with the mnemonic displayed in parentheses following the text of the menu option. If all of the characters in a menu option have been assigned as mnemonics or if the choice consists of multi-byte characters, the application can use another letter or keyboard character. The same mnemonic is assigned to an option whenever it appears in an application menu.

The layout of the user's keyboard needs to be considered when selecting the key combinations for the mnemonics and accelerators to assign to translated menu options. First, keys are selected to minimize the disruption or relearning required to execute a mnemonic or accelerator, especially a frequently used one. Second, there are no conflicts between the key combinations for entering accented characters (e.g., if a US keyboard is being used) and those being used for mnemonics and accelerators. Finally, some non-US keyboards contain only one <Alt>, located either on the left or right side of the

keyboard. The ease with which users can execute the key combination for a mnemonic is considered if one of these keyboards is being used.

Speed search and text search. If the application is translated into a language that contains accents on the first letter of words, users are able to execute a speed search (e.g., in list boxes) by typing an unaccented upper-case or lower-case letter, and the search finds instances of both unaccented and accented first letters.

If the application uses wild card characters (i.e., @, #, ?, and *) to perform text searches, it needs to determine if these characters are assigned special meaning in the target language. If necessary, an alternate set of wild card characters is selected to eliminate any possible confusability when the application is converted to the target language.

12.4.6 Text Manipulation

Sorting and collation. Applications make use of linguistic sort sequences to order the contents of alphanumeric lists or to add new information to an already sorted list. In US software, characters are usually compared according to their binary value in the code set, with characters ordered based on these values. However, variations are frequently required to reflect linguistic conventions since the binary sequence of characters may not match the linguistic sequence for the language. For example, variations may be needed to handle characters with functional equivalence (e.g., Mac and Mc usually appear together) and to address situations where a character should be ignored (e.g., re-locate and relocate should be placed together). In addition, where US software typically provides a single sorting algorithm to accommodate such variations, other languages usually support multiple sort orders. As a result, the application needs to provide users with the ability to choose a sort order that meets their needs.

Sorting rules for European languages must be able to handle extended character sets and language-specific conventions, independent of the binary values assigned to characters. These languages may contain letters after "z" or sort letters out of the standard alphabetic sequence used in the US. For example, some of these languages contain double characters that sort as one combined character, or a single character that is treated as a double character. In Spanish, double characters such as "ch" and "ll" sort as a single character, and in German, ß is a single character that is treated as "ss" when found in a word. Madell, Parsons, and Abegg provide the following examples of differences in sorting order based on ASCII and German rules, and ASCII and Spanish rules:

Sorted by ASCII rules	Sorted by German rules	Sorted by ASCII rules	Sorted by Spanish rules
Airplane	Airplane	chaleco	cuna
Zebra	ähnlich	cuna	chaleco
bird	bird	día	día
car	car	llave	loro
ähnlich	Zebra	loro	llave
		maíz	maíz

In the case of complex (i.e., multi-byte) languages, expressions can be written in a mixture of character sets. For example, the Japanese word for "water" may be written as a single Kanji character, as two Hiragana characters, as two Katakana characters, or as the four-letter Romaji expression "mizu." As a result, sorting algorithms in these languages must be able to accept multiple character patterns as representing the same expression. These algorithms can combine a sorting order among the character sets with a sorting order for expressions within each set. In addition, the application may need to provide a sort order based on a symbol feature that is not captured within the character code. For example, Chinese expressions may need to be sorted by the numeric value of the character as represented in the coded character set as well as by the number of strokes required to represent the

character, the radical (i.e., root) of the character, or the number of strokes added to the radical. Finally, the application may need to implement a sort order based on the way symbols are pronounced. In this case, each symbol may have to be stored in both graphic and phonetic form, with the resulting sort order listing symbols that are phonetically similar but visually different near each other.

Editing functions. Editing functions (e.g., search and replace, cut and paste, and spell checking) can accommodate the unique features of the target language, including instances where the appearance of a word changes when it is hyphenated, where it appears in lower rather than upper case, or where it contains a combination character such as ⁋. In contextual languages such as Thai, the characters that make up a compound symbol may be entered in several different orders, with the appearance of the symbol varying based on the order in which the characters are entered. In other languages (e.g., Greek), the appearance of a character can vary depending on its position in a word. If the application performs string searches in these languages, it is able to recognize any of several possible character sequences and judge them to be the same or different as appropriate.

12.4.7 Adjustments for Bidirectional Languages

If the application is being converted to a bidirectional language such as Hebrew or Arabic, it implements a right-to-left screen orientation. Window appearance for these languages is the mirror image of the corresponding US (i.e., left-to-right-oriented) window, except that the placement of the Window menu, Maximize, and Minimize buttons in the title bar does not change. Window titles, headings, messages, and controls are translated, except for English acronyms and key names (e.g., F1, Alt) and key combinations (e.g., Shift+Del).

In general, all of the information in a window is displayed in an orientation that is correct for the user. With respect to information placement, the specifications in this style guide apply, except that "right" and "left" are interchanged. However, physical right and left remain the same. As in unidirectional languages, <Left> and <Right> move the cursor in the direction of the arrow indicated on the key; the right and left buttons on the pointing device behave as defined in this document; and left and right movement of the pointing device moves the pointer in these directions. If the application chooses to mix both right-to-left and left-to-right elements within the same window, it follows the relevant specifications defining information display for unidirectional and bidirectional languages.

12.4.8 Printing

Peripheral devices such as printers are capable of handling the character set for the target language. The full character set can be loaded on the printer, and the printer can produce all of the extended characters required by the language.

While the US standard paper size is 8.5 x 11 inches, most countries use ISO A4 size which is slightly longer and narrower than the US standard. As a result, printer capabilities (e.g., different paper trays) may need to be adjusted in order to handle the standard paper and envelop sizes used by the target audience, and the application may need to be modified to handle the varying page layouts dictated by the different paper sizes. For example, hardcoded rules regarding paper margins are removed, and users are allowed to specify how they want text to appear and to do so using measurement units with which they are familiar.

Adjustments made in window format to accommodate text expansion also need to consider text presentation when the content of the window is printed. In particular, the amount of vertical space between lines of text is sufficient to print all extended characters, including those with accents, in both upper and lower case.